JP6266553

Publication Title:

COMPUTER SYSTEM

Abstract:

Abstract of JP6266553

PURPOSE:To support a CISC type instruction by an RISC type processor. CONSTITUTION:A conversion table 2 stores the correspondence between the instruction codes that are not supported by a CPU 1 and a string of instruction codes that are supported by a processor and attains a function equivalent to those instruction codes that are not supported by the CPU 1. If the instruction code supplied to the CPU 1 directly from a memory 7 or via an instruction cache 4 is not supported by the CPU 1, a controller 5 successively supplies the instruction codes forming a string of instruction codes stored in a conversion table to the CPU 1 in place of those instruction codes supplied from the memory 7.

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報(A)　(11)特許出願公開番号

特開平6－266553

(43)公開日　平成６年(1994)９月22日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　9/30	３１０　Ｅ	9189－5Ｂ		

審査請求　未請求　請求項の数３　ＯＬ　（全６頁）

(21)出願番号　特願平5－56102

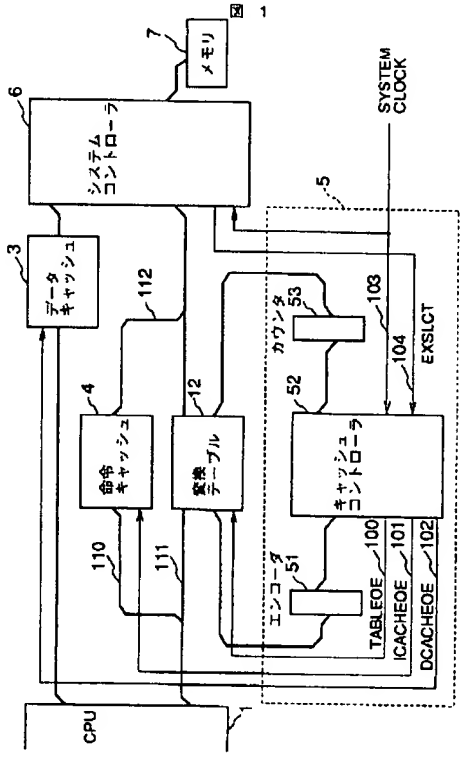
(22)出願日　平成５年(1993)３月16日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　筒井　基次
神奈川県海老名市下今泉810番地　株式会社日立製作所オフィスシステム事業部内
(74)代理人　弁理士　富田　和子

(54)【発明の名称】　計算機システム

(57)【要約】
【目的】ＲＩＳＣ型プロセッサで、ＣＩＳＣ型命令をサポートする。
【構成】変換テーブル２は、ＣＰＵ１がサポートしていない命令コードと、前記プロセッサがサポートする命令コードの列であって、前記ＣＰＵ１がサポートしていない命令コードと等価な機能を果たす命令コードの列との対応を記憶する。メモリ７より直接もしくは命令キャッシュ４を介して、ＣＰＵに供給された命令コードが、ＣＰＵがサポートしていない命令コードであった場合、コントローラ５は、前記メモリ７より供給された命令コードに代えて、当該命令コードに対応して前記変換テーブルに記憶されている命令コードの列を構成する各命令コードをＣＰＵ１に順次供給する。

図 1



1

【特許請求の範囲】

【請求項1】プロセッサと、プロセッサに供給する命令を格納したメモリとを備えた計算機システムであって、

前記プロセッサがサポートしていない命令コードに対応させて、前記プロセッサがサポートする命令コードの列であって、前記プロセッサがサポートしていない命令コードと等価な機能を果たす命令コードの列を記憶した変換テーブルと、

前記メモリより前記プロセッサに供給された命令コードが、前記プロセッサがサポートしていない命令コードであった場合に、前記メモリより供給された命令コードに代えて、当該命令コードに対応して前記変換テーブルに記憶されている命令コードの列を構成する各命令コードを前記プロセッサに順次供給する制御手段とを有することを特徴とする計算機システム。

【請求項2】請求項1記載の計算機システムであって、

前記変換テーブルは、前記プロセッサより書換え可能であることを特徴とする計算機システム。

【請求項3】請求項1記載の計算機システムであって、

前記変換テーブルは、プロセッサがサポートしていない命令コードによって指定されるアドレスより順次、前記プロセッサがサポートする命令コードの列であって、前記プロセッサがサポートしていない命令コードと等価な機能を果たす命令コードの列と、当該命令コードの列の最終の命令コードの識別するための識別情報を記憶し、

前記制御手段は、メモリより前記プロセッサに供給された命令コードが、前記プロセッサがサポートしていない命令コードであった場合に、当該命令コードによって指定されるアドレスより、前記識別情報で指定される当該命令コードの列の最終の命令コードまで、前記変換テーブルに記憶されている命令コードを、順次読みだして、前記プロセッサに供給することを特徴とする計算機システム。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、ＲＩＳＣ（Reduced Instruction Set Computer)型プロセッサを用いた計算機システムに関し、特に、そのエミュレーションの技術に関するものである。

【0002】

【従来の技術】近年のワークステーションにおいて用いられるＭＰＵ（Micro proccessor unit）は、ＲＩＳＣ型が主流となりつつある。しかし、ＲＩＳＣ型アーキテクチャーの採用により、ハードゥエア性能、機能は著しい発展をみせている反面、このようなＲＩＳＣ型プロセッサの性能を最大限に引き出すために必要となるコンパイラ技術等のソフトウェア技術の開発の遅れが問題となってきている（日経エレクトロニクス１９９０．６．１１．Ｐ１７２）。

【0003】また、新たにＲＩＳＣ型プロセッサを採用

2

したコンピュータを製作する場合等、ＣＩＳＣ型プロセッサを採用していた従来の機種との互換性の維持することが難しい（日経エレクトロニクス１９９２．４．２７．Ｐ１１５）。

【0004】

【発明が解決しようとする課題】さて、ソフトウェアの開発工数を減らすためには、従来の蓄積されたノウハウの活用ができることが一番望ましい。特に、従来の機種のバイナリレベル（マシン語レベル）の命令セットを、そのまま新たな機種に流用可能となればより高い移植性を実現でき、また製品の互換性が保つことができる。

【0005】ところで、ＣＩＳＣ型プロセッサは、様々な処理サイクルの複雑な命令を含む命令セットを備えている。したがい、少ないステップ数のプログラムを作成するこができ、またソフトウェアの開発工数も削限できる。一方、ＲＩＳＣ型プロセッサは複雑な命令をサポートせずに、命令を同じ処理サイクルで実行できるものにのみに制限することで、スムーズなパイプライン処理を実現することにより処理の高速化をはかっている。

【0006】したがい、ＣＩＳＣ型プロセッサ上で開発したマシン語レベルのソフトウェアは、ＲＩＳＣ型プロセッサで実行することはできない。

【0007】すなわち、ＣＩＳＣ型プロセッサを用いたコンピュータと、ＲＩＳＣ型プロセッサを用いたコンピュータの間には互換性がない。

【0008】このため、ＣＩＳＣ型プロセッサを用いたコンピュータ上で開発したソフトウェア資産や、蓄積されたノウハウを、ＲＩＳＣ型プロセッサを用いたコンピュータ上で利用することができない。

【0009】そこで、本発明は、ＲＩＳＣ型プロセッサにおいて、そのパイプライン処理を乱すことなく、ＣＩＳＣ型プロセッサや他のプロセッサの命令をサポートすることのできるアーキテクチャを提供することを目的とする。

【0010】

【課題を解決するための手段】前記目的達成のために、本発明は、プロセッサと、プロセッサに供給する命令を格納したメモリとを備えた計算機システムであって、前記プロセッサがサポートしていない命令コードに対応させて、前記プロセッサがサポートする命令コードの列であって、前記プロセッサがサポートしていない命令コードと等価な機能を果たす命令コードの列を記憶した変換テーブルと、前記メモリより前記プロセッサに供給された命令コードが、前記プロセッサがサポートしていない命令コードであった場合に、前記メモリより供給された命令コードに代えて、当該命令コードに対応して前記変換テーブルに記憶されている命令コードの列を構成する各命令コードを前記プロセッサに順次供給する制御手段とを有することを特徴とする計算機システムを提供する。

【0011】

【作用】本発明に係る計算機システムによれば、たとえば、メモリよりＲＩＳＣ型プロセッサに供給された命令コードが、前記プロセッサがサポートしていないＣＩＳＣ型命令コードであった場合にも、これに代えて、当該命令コードに対応してあらかじめ、変換テーブルに記憶しておいた、当該ＣＩＳＣ型命令と等価な機能を果たすＲＩＳＣ型命令コードの列を構成する各ＲＩＳＣ型命令コードを前記ＲＩＳＣ型プロセッサに順次供給するころができる。

【0012】よって、アプリケーションプログラムの依存するプロセッサの型や種別によらずに、当該アプリケーションプログラムを実行することができる。また、この際、ＲＩＳＣ型プロセッサに与えられる命令はＲＩＳＣ型命令のみであるので、そのパイプライン処理が乱れることもない。

【0013】

【実施例】以下、本発明の一実施例を説明する。

【0014】図１に、本実施例に係る計算機システムの構成を示す。

【0015】図中、１はＲＩＳＣ型のＣＰＵ、２は変換テーブル、３はデータキャッシュ、４は命令キャッシュ、５はコントローラ、６はシステムコントローラ、７はＤＲＡＭで構成されるメモリである。また、コントローラ５は、キャッシュコントローラ５２と、エンコーダ５１と、カウンタ５３を内蔵した１チップコントローラＩＣとして実現されている。

【0016】ＣＰＵ１は、所望のデータをデータキャッシュ３より、所望の命令を命令キャッシュ１４より取り込み処理する。所望の命令やデータが命令キャッシュ４もしくはデータキャッシュ４上に存在しない場合には、システムコントローラによってメモリ上の命令、データが、それぞれ命令キャッシュ４、データキャッシュにロードされる。各部は、システムクロック１０３に同期して動作する。

【0017】さて、ＣＰＵ１は、システムの起動時等に、システムコントローラ６内のレジスタに、命令セットの変換を行うか否かを指定する情報をセットする。システムコントローラ６は、変換を行わない旨が指定された場合には、信号１０４をネゲートする。信号１０４がネゲートされると、コントローラ５のキャッシュコントローラ５２は、信号１００をネゲートすることにより変換テーブルのパス１１１への出力を抑止する。一方、信号１０１をアサートし、命令キャッシュ２からＣＰＵ１へのパス１１０による直接出力をイネーブルにする。このように、命令セットの変換を行わないモードでは、変換テーブル２が無効化された状態であり、従来の計算機システムと同様に動作する。

【0018】一方、変換を行う旨が指定された場合には、システムコントローラ６は、信号１０４をアサートする。信号１０４がアサートされると、コントローラ５のキャッシュコントローラ５２は、信号１００をアサートすることにより変換テーブルのパス１１１への出力を許可する。一方、信号１０１をネゲートし、命令キャッシュ２からＣＰＵ１へのパス１１０による直接出力を抑止する。パス１１０による出力が抑止されると、命令キャッシュ４からＣＰＵ１への出力される命令は、パス１１２を介して変換テーブル２に供給される。

【0019】変換テーブル２はＳＲＡＭにより構成され、命令キャッシュとＣＰＵの間に配置される。コントローラ５とこの変換テーブル２は、命令キャッシュから、ＣＰＵ１に取り込む命令がＣＩＳＣ型の命令である場合には、これをＲＩＳＣ型の命令列に変換する。

【0020】ここで、図２に、変換テーブル２で行うＣＩＳＣ型命令の、ＲＩＳＣ型命令列への変換の関係を示す。

【0021】図２のＣＩＳＣ型命令”ＡＤＤ　メモリ、レジスタ１”は、メモリの値をレジスタ１の値と加算し、元のメモリに結果を渡す加算命令である。

【0022】ＲＩＳＣ型のプロセッサであるＣＰＵ１は、このようなメモリに対する演算命令をサポートしていない。そこで、変換テーブル２において、ＣＩＳＣ型命令”ＡＤＤ　メモリ、レジスタ１”と同じ結果を得ることのできる、３つのＲＩＳＣ型命令よりなる命令列にＣＩＳＣ型命令”ＡＤＤ　メモリ、レジスタ１”を変換する。

【0023】すなわち、演算されるメモリの値をレジスタ２にロードする命令”ＬＯＤ　レジスタ２、メモリ”と、レジスタ１、２間の加算を行い結果をレジスタ１に格納する命令”ＡＤＤ　レジスタ１　レジスタ２”、そして、レジスタ１の内容を元のメモリにストアする命令”ＳＴＡ　メモリ、レジスタ１”の３つのＲＩＳＣ型命令の命令列に、ＣＩＳＣ型命令”ＡＤＤ　メモリ、レジスタ１”を変換する。

【0024】また、たとえば、図３に示したＣＩＳＣ型命令”ＡＤＤ　メモリ１　メモリ２”は、メモリ上のメモリ１というアドレスの内容に、メモリ上のメモリ１というアドレスの内容を加算する命令であるが、この命令については、４つのＲＩＳＣ型命令よりなる命令列”ＬＯＤ　レジスタ１　メモリ１”、”ＬＯＤ　レジスタ２　メモリ２”、”ＡＤＤ　レジスタ１　レジスタ２”、”ＳＴＡ　メモリ２レジスタ２”に変換するようにする。

【0025】このように、本実施例に係る計算機システムでは、ＣＰＵ１に取り込む命令がＣＩＳＣ型の命令である場合には、これを同等の機能を果たすＲＩＳＣ型の命令列に変換する。

【0026】以下、命令コードが１６ビットとして、図２で示した変換動作を例にとり、このようなＣＩＳＣ型命令からＲＩＳＣ型命令列への変換動作について、図４

5

を用いて説明する。

【００２７】この場合、図１の変換テーブル２は、２０ビット幅のアドレスを有するＳＲＡＭで構成する。

【００２８】いま、ＣＰＵから命令を要求されると、コントローラ５は命令キャッシュ４もしくはシステムコントローラ８に命令フェッチを要求し、命令を一つ読みだす。読みだされた命令の各ビットは変換テーブル２を構成するＳＲＡＭのアドレスの上位１６ビットに入力される。一方、コントローラ５は、カウンタ５３より、ＳＲＡＭのアドレスの下位４ビットとして“０ｈ”を出力する。

【００２９】今、入力された命令がＣＩＳＣ型加算命令１２５４ｈである場合、ＳＲＡＭのアドレスは、コントローラにより生成される下位４ビット“０ｈ”とあわせて、１２５４０ｈとなる。

【００３０】このアドレスには、あらかじめ、ＣＩＳＣ型加算命令加算命令と同じ結果を得ることのできる、３つのＲＩＳＣ型命令よりなる命令列の第１番目のＲＩＳＣ型命令ＬＯＤ；１００１ｈの下位側に、この命令列の命令数の残りを表す４ビットを連結した値を記憶させておく。この場合、残りの命令数は２個であるので、アドレス１２５４０ｈには、１００１２ｈを記憶しておく。

【００３１】ＳＲＡＭのアドレス１２５４０ｈより出力された値１００１２ｈの上位１６ビット１００１ｈは、ＣＰＵ１に命令として与えられる。ＣＰＵ１は、この命令を実行し、コントローラ５に次の命令を要求する。一方、ＳＲＡＭのアドレス１２５４０ｈより出力された値１００１２ｈの下位４ビット２ｈは、コントローラ５に入力される。コントローラは、エンコーダ５１により、入力された値を判定し、入力された値が”０ｈ”でない場合、ＣＰＵから命令要求を受け取ると、カウンタ５３を１インクリメントし、ＳＲＡＭのアドレスの下位４ビットを１インクリメントする。また、これによりシステムコントローラ６のウェイトの調整を行う。

【００３２】これにより、ＳＲＡＭのアドレスは１インクリメントされ、アドレス１２５４１ｈが入力される。ＳＲＡＭのこのアドレスには、あらかじめ、ＣＩＳＣ型加算命令加算命令と同じ結果を得ることのできる、３つのＲＩＳＣ型命令よりなる命令列の第２番目のＲＩＳＣ命令ＡＤＤ；２０１０ｈの下位側に、この命令列の残りの命令数１を表す値１ｈを連結した値、２０１０１ｈを記憶させておく。

【００３３】これにより、値２０１０１ｈの上位１６ビットの命令ＡＤＤ；２０１０ｈは、ＣＰＵ１に命令として与えられる。ＣＰＵ１は、この命令を実行し、次の命令をコントローラ５に要求する。

【００３４】一方、ＳＲＡＭのアドレス１２５４１ｈより出力された値の下位４ビットを受け取り、これが”０ｈ”でないので、先程と同様に、ＣＰＵから命令要求を受け取ると、ＳＲＡＭのアドレスの下位４ビットを１インクリメントする。

6

【００３５】これにより、今度は、ＳＲＡＭのアドレスは１インクリメントされ、アドレス１２５４２ｈが入力される。ＳＲＡＭのこのアドレスには、あらかじめ、ＣＩＳＣ型加算命令加算命令と同じ結果を得ることのできる、３つのＲＩＳＣ型命令よりなる命令列の第３番目のＲＩＳＣ命令ＳＴＡ；１００２ｈの下位側に、この命令列の残りの命令数０を表す値０ｈを連結した値、１００２０ｈを記憶させておく。

【００３６】これにより、値１００２０ｈの上位１６ビットの命令ＳＴＡ；１００２ｈは、ＣＰＵ１に命令として与えられる。ＣＰＵ１は、この命令を実行し、次の命令をコントローラ５に要求する。

【００３７】一方、コントローラは、ＳＲＡＭのアドレス１２５４０ｈより出力された値の下位４ビットを受け取り、これが”０ｈ”であるので、カウンタ５３をリセットし、ＳＲＡＭのアドレスの下位４ビットを０ｈに戻し、次の命令フェッチを、命令キャッシュ４またはシステムコントローラ６に対し要求し、次の命令を読みだす。

【００３８】さて、命令キャッシュ４またはシステムコントローラ６から入力された命令が、ＲＩＳＣ型命令である場合には、この命令をこのまま、スルーでＣＰＵ１に渡せばよいのであるが、アーキテクチャを簡略化する為、本実施例では、ＳＲＡＭ内にＲＩＳＣ型命令に対応する値も記憶しておく。

【００３９】すなわち、図５に示すように、ＲＩＳＣ型のロード命令ＬＯＤ；１００１ｈが入力された場合も、ＣＩＳＣ型命令の場合と同様に、命令の各ビットをＳＲＡＭのアドレスの上位１６ビットに入力し、コントローラ５は、ＳＲＡＭのアドレスの下位４ビットとして“０ｈ”を出力する。

【００４０】これによって得られるＳＲＡＭのアドレス１００１０ｈには、あらかじめ、このＲＩＳＣ命令ＬＯＤ１００１ｈの下位側に、０ｈ連結した値を記憶させておく。　これにより、値１００１０ｈの上位１６ビットの命令ＬＯＤ；１００１ｈは、ＣＰＵ１に命令として与えられる。ＣＰＵ１は、この命令を実行し、次の命令をコントローラ５に要求する。

【００４１】一方、コントローラは、ＳＲＡＭのアドレス１００１０ｈより出力された値の下位４ビットを受け取り、これが”０ｈ”であるので、ＳＲＡＭのアドレスの下位４ビットを０ｈのままとし、次の命令フェッチを、命令キャッシュ４またはシステムコントローラ６に対し要求し、次の命令を読みだす。

【００４２】なお、変換テーブル２の、ＣＩＳＣ型命令に対応する部分の内容は、システムイニシャライズ時にＣＰＵが設定するができる。また、システムイニシャライズ時以外のときにも、適宜に書換えることにより、様々な命令セットに対応することがでいるようになる。な

7

お、イニシャライズプログラムは、ＲＩＳＣ型命令によって記述する。

【００４３】なお、以上の実施例では、ＣＩＳＣ型命令コードによって、直接指定されるアドレスから順次、変換テーブルに当該ＣＩＳＣ型命令と等価な機能を果たすＲＩＳＣ型命令コードの列を記憶したが、これは、かならずしも、ＣＩＳＣ型命令コードによって直接指定されるアドレスから順次記憶する必要はなく、任意のアドレスから記憶し、記憶したアドレスと対応するＣＩＳＣ型命令コードの対応を記憶させた適当なアドレス変換テーブルを用いて、処理を行うようにしてもよい。

【００４４】以上のように、本実施例によれば、命令コードを自由に換えられるため、異なる機種間のソフトウェアの移植が容易となる。そのためソフトウェアの開発工数を削減することが可能となる。また、ＲＩＳＣ型ＣＰＵが実行する命令は、ＲＩＳＣ型命令のみであるので、そのパイプライン処理が乱れることもない。

【００４５】

【発明の効果】以上のように、本発明によれば、ＲＩＳＣ型プロセッサにおいて、そのパイプライン処理を乱すことなく、ＣＩＳＣ型プロセッサや他のプロセッサの命令をサポートすることができる。

8

【図面の簡単な説明】

【図１】本発明の一実施例に係る計算機システムの構成を示すブロック図である。

【図２】本発明の一実施例に係る計算機システムの行う命令の変換の第１の例を示す説明図である。

【図３】本発明の一実施例に係る計算機システムの行う命令の変換の第２の例を示す説明図である。

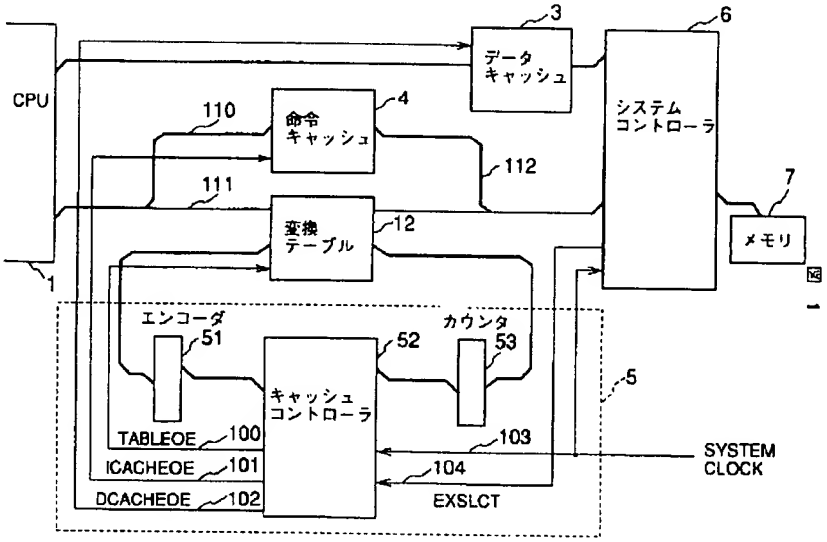
【図４】本発明の一実施例に係る計算機システムの行う命令の第１の変換動作を示す説明図である。

【図５】本発明の一実施例に係る計算機システムの行う命令の第２の変換動作を示す説明図である。

【符号の説明】

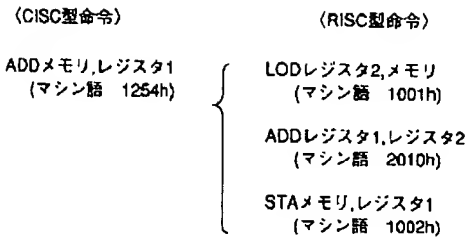
１　ＲＩＳＣ型のＣＰＵ
２　変換テーブル
３　データキャッシュ
４　命令キャッシュ
５　コントローラ
６　システムコントローラ
７　メモリ
５１　エンコーダ
５２　キャッシュコントローラ
５３　カウンタ

【図１】



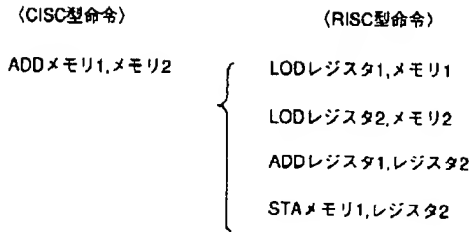
【図2】

図 2



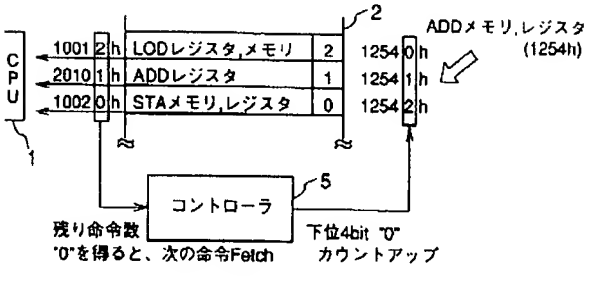
【図3】

図 3



【図4】

図 4



【図5】

図 5